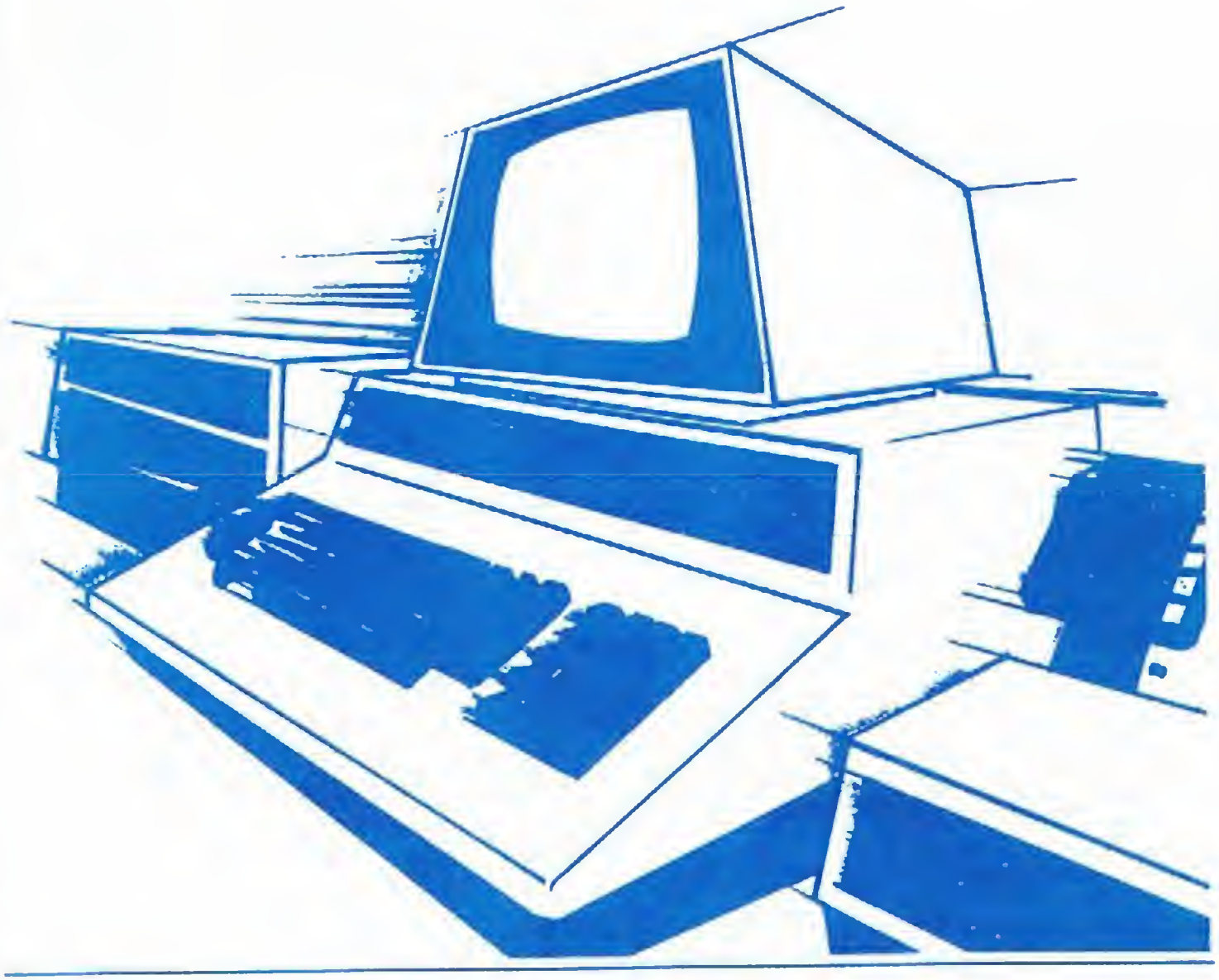
CPUCN

The Official Commodore Pet Users Club Newsletter



Volume 2 Issue 3

commodore

Contents

Editorial

Commodore News — new 8K PETs; even newer PETs; Training Program '80

Software Review — Commodore's Work Processor II — New cassette releases — Space Invaders

New/Old ROM Memory Map — Conversion Tables

Upper/Lower Case Converter (for programs written on old ROM PETs)

Trace

Supermon

Programming Tips

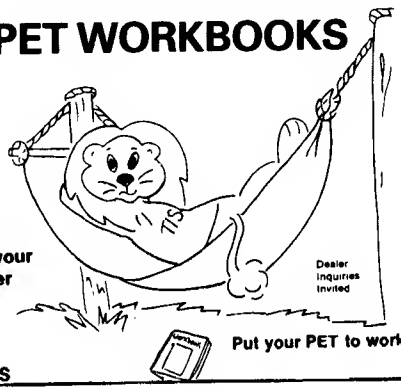
For/Next Loop Structures

Duplicating Cassettes for Commodore

Job Vacancies

PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET

PET WORKBOOKS



See your Dealer

Dealer inquiries invited

Put your PET to work!

T I S

WB-1 Getting Started with Your PET
WB-2 PET String and Array Handling
WB-3 PET Graphics
WB-4 PET Cassette Input/Output
WB-5 Miscellaneous PET Features
WB—6 PET Control and Logic

£18 for the full set

From your dealer or direct from:

ACT Petsoft

66-68 Hagley Road, Edgbaston, Birmingham B16 8PF

(Please add 75p to cover postage and packing.)

PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET • PET

Editorial

From all of us in Commodore our best wishes to all our readers for a Happy Christmas and a successful New Year. We hope that, the Post Office willing, all but some of our overseas subscribers will receive this issue of CPUCN within a few days of the Christmas holiday.

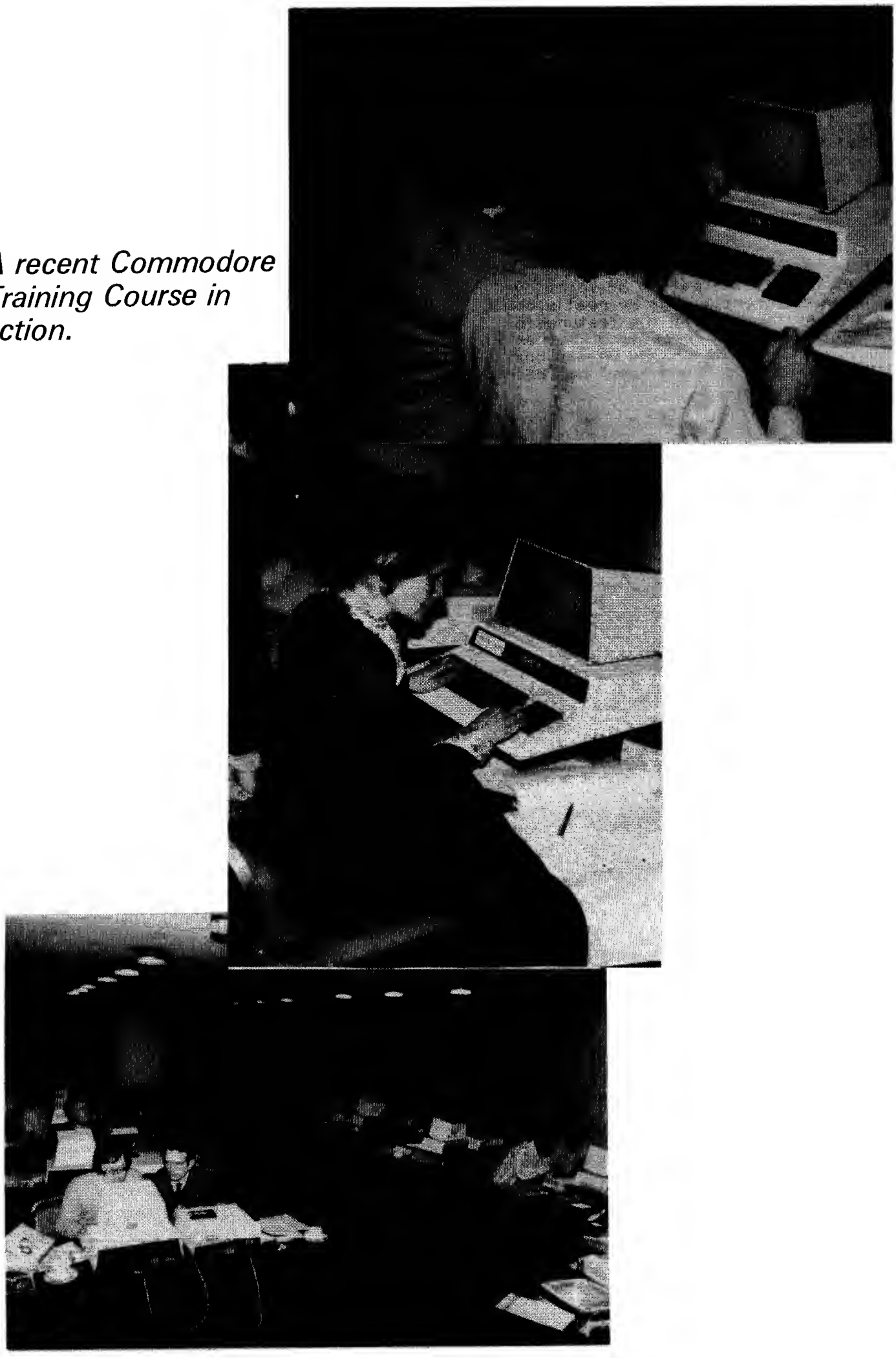
We have changed the page layout somewhat, in anticipation of going over to a typeset format. But we really could not resist showing off the features of the Commodore Word Processing Program II and the 3022 printer - and we can boast that this issue's "hard copy" has been produced entirely on Commodore equipment.

The "MACHINE CODE FOR BEGINNERS" course has been held over until the next issue. We are using the space to publish SUPERMON which previews some of the advanced features available in the official Commodore Assembler package. SUPERMON will be used in future sections of the machine code course.

See you again in the New Year!

Andrew Goltz

A recent Commodore Training Course in action.



Commodore News

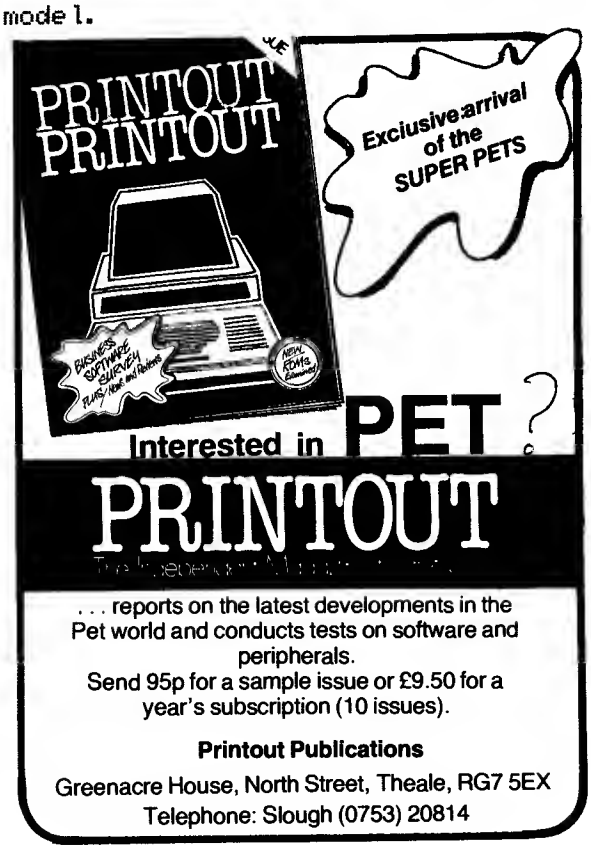
NEW PETS!

A brand new 8K PET with all the advantages of the new ROM operating system AND a large keyboard has joined the ranks of the Commodore PET range.

Introduced as a result of the interest shown by many 8K owners in the possibility of adding a large keyboard, the new PET is a further example of Commodore's philosophy of providing you with the products that YOU want at a price which gives you the best possible value for money.

The new PETs are ideally suited to the needs of Education, the Laboratory, Industrial Control, and the Home User. The large keyboard takes up the space previously available for a built-in cassette deck. However, with the new PET priced at $495 (+Vat), a configuration offering all the advantages of the new 8K PET, and including one external cassette deck will only cost as much as the old 8K model. In a multi-PET environment, where a number of machines can be loaded from a single cassette deck, substantial cost savings can be made.

The new PETs incorporate the new dynamic RAM chips which have proved themselves ultra reliable in the 16K and 32K machines. As the new ROMs are included as a standard feature owners wishing to add a floppy disk unit at a future date will be able to simply "plug in and go", with no need to "retrofit" a different set of ROMs as was the case with the old 8K model.



MORE NEW PRODUCTS ON THE WAY!

With Commodore's Research and Development budget running at a staggering $5m a year we hear through "the grapevine" of a whole range new products currently in the research labs. Larger capacity disks, Larger capacity PETs, Colour PETs!!!, pocket calculator sized PETs, intelligent MODEMs (the interest shown in our research work by our competitors nessecitates that we draw a veil over our research activities at this point. Watch this space for futher details! - Ed)

BIGGER TRAINING PROGRAM IN 1980

As a result of the interest shown in our 1979 pilot Training Program, Commodore will be extending the range of training course available in 1980.

All except the most proficient programers will find it useful to attend the new ADVANCED BASIC course. This course will introduce a numbers of advanced programming techniques, and will also draw attention to a number of "bad habits" which can creep into the work of even quite experienced programmers.

Those interested in using the PET for monitoring scientific equipment or industrial control will find the new MONITORING & CONTROL seminars of particular value. These are being run by us and supported by a team of Commodore dealers with specialist experience in this field, including - MACHSIZE (who recently developed a bar code reader for the PET); ANASPEC (who offer the PET interfaced to a wide range of analytical equipment); and PETALECT (who pioneered the hook-up of electronic balances to the PET).

A series of seminars is also being run for those contemplating the introduction of a microcomputer into their business. The speakers will include representatives from Commodore, a Business Software dealer, and a Business User, who will talk about his own experiences. So if any of your business friends are looking at the possibility of purchasing a micro, after you have thouroughly confused them with SPACE INVADERS and 3D STARTREK, send them along to the COMMODORE BUSINESS SEMINARS!

A leaflet containing full details of our new training program should be included with this newsletter. If it was accidently missed out, please contact the Commodore Information Centre at 360, Euston Road, London NW1, for full details.

Software Review

COMMODORE WORD PROCESSOR II

- Mike Whitehead -

In this issue I thought I would continue the process of describing the new Commodore disk software releases by giving you a brief introduction to the Commodore Word Processor II.

First I should mention that this program (BS1100) is in the Commodore Business Software library, and as such can only be purchased from that subset of Commodore dealers who are also Commodore Business Software dealers. The idea is that the more complicated business programs may well take a good deal of demonstrating, explaining, installing, and continued support in the field. Therefore we only release these programs through dealers we feel capable of performing these functions effectively. All this being said, however, WordPro II isn't exactly the hardest program to become familiar with, given that you are somewhat used to computers.

To start with, Word Pro II comes with a manual, a diskette with two different versions of the program and two manuals on it, a Master Security ROM, and a big white binder to hold it all. The package costs $75.00, and anyone who has seen it in operation will agree that it is an outstanding value. Your dealer must insert the Master Security ROM in your 16K or 32K PET before any of the disk-based Business Software programs will work. This approach has been taken to prevent illegal copying of copyright software. Our authors have invested a great deal of time and money to produce our business packages, and we feel obliged to take any and all steps we can to protect that investment - as Pet users we ask you to help in this.

The two manuals which are available for WordPro II are the reference manual and the training manual. WordPro files for both manuals are provided on your diskette, and a printed copy of the reference manual is also included. For non computer types, the training manual is ideal. It was written by Mike Gross-Niklaus, who is our training manager here at Commodore. Mike has a gift for explaining almost anything in very straight-forward terms - so certainly most typists should be able to learn WordPro by using the training manual.

Once you have got your Security ROM installed and have read the manual, you are ready to start word processing. If you have a Commodore printer then you LOAD and RUN the program CBM EDITOR. If you have a printer which operates on the standard ASCII character set, then you use the program ASC EDITOR. Both are complete versions of the WordPro program. Memory space within the PET is divided into Main text and Extra text, as defined by WordPro. Generally speaking these two areas are interchangeable as they both are used to store text, but in practice Main text is used to store the body of a letter and Extra text holds any special insertions it may need. The dividing line in memory between Main text and Extra text may be moved, to give you less of one and more of the other. In fact the very first question you are asked after running WordPro is how many lines of text you wish to allocate to Main text, with the remainder going to Extra text.

Once you get past that question the screen comes up with the standard display for the WordPro environment. This consists of a status line which reads:

PET Text Editor :X:I:S:C:N: C= 1 L= 1

What does all this mean then? Let's start at the right-hand side and work backwards. The C and L values correspond to the current line and character position of the cursor. As the entire package is written in machine code, it is no problem at all for these numbers to be updated in real time, as you type in. Next comes all the letters surrounded by colons. These are status indicators which will light up (reverse characters) if any optional mode of operation has been selected. X stands for Extra text, and is to remind you when that is your current text environment. I stands for Insert mode. If you position the cursor in the middle of a word and enter insert mode (SHIFT OFF/RVS) then anything you type in will automatically push any text after it further to the right and/or onto the next line. S stands for shift lock. For this you don't use the normal shift lock key, but rather the back slash key located on the top row of the keyboard. C is used to indicate control mode, meaning that you are in the middle of some special function initiated by the control key (sorry, it's the OFF/RVS key!). N is a rather special indicator. It's three values are N, U, and L, for Uppercase, Lowercase, and Normal. These three states correspond to the transformation performed on the characters on the screen when the cursor is passed over them from left to right. To transform 'lowercase' into 'LOWERCASE' you type Control U (to set into uppercase translate mode) and pass the cursor from left to right over the word in question. The last bit displayed on the status line is PET Text Editor, which is sometimes replaced with other text in various special modes.

Now that I've got all of this nasty overhead out of the way, let's see what it is like to operate the system in practice. First off let me explain that WordPro consists of two major portions - one that let's you type in text, modify and review it, and the other portion which let's you type the document out. Within output mode (entered by control O) the screen display changes entirely, displaying only a menu of current output options - such as margin settings, spacing, pagination, and right justification. Once you have these options set as desired, you are ready to print. However there is not only one print command. T for type is the usual one, for simple printing, but a few others exist to allow you to type several copies of a given letter at once, or to type several copies of a variable insertion form letter at once. This last capability is one of the very most powerful available to you within WordPro, and will be dealt with in depth shortly. For the meantime, let's go back to the typing/editing environment (E to exit back from output mode).

You can type anything you like, anywhere you like on the screen, and have a software enabled repeat key available to you for moving the cursor around quickly. Input is free format, in that words wrap around from line to line - the document only takes its final shape when it is printed.

There is no need to hit the return key at the end of each line - just type continuously and WordPro will look after the formatting when it is time to print. Returns are only used to force new lines of print (such as at the end of a paragraph) or to leave blank lines in the printed text. When you think your document is about right, you can print it out to check. If it needs correcting you have the normal insert and delete key functions available, and a few special capabilities as well. Special editing functions include paragraph move, line insert, line delete, block insert, and string search. When the document is corrected you may save it on the disk if you think you will want it again. Disk functions available include Load, Save, Directory, and an environment rather like DOS Support.

The Load function even lets you do an Append!

WordPro II runs on both 16K and 32K Pets, but will only ever use 16K of memory. As the program itself is about 8K long, the user has 196 screen lines open to him - each 40 characters wide. At most 173 of these lines may be devoted to the main text area, but this is no real restriction as any given file should only ever include one printed page of text anyway. This orientation explains why WordPro II is best used for letters and short documents - each page of print must be recalled from disk and printed as separate operations, which could get a bit tedious after too many pages.

One really powerful capability of the WordPro package that I haven't said much about is form letter writing. A typical form letter might be about half a page of A4, and contain 4 or 5 spots where you would like the letter to be personalized. Name, address, date, amount due, items and quantities shipped - these are all good candidates for variable insertion blocks, the points at which WordPro letters may be personalized. If you are typing in the skeleton of a form letter and come to a spot which varies, you only need type Control B. This leaves an odd mark on the screen to remind you of the insertion point. When you have finished the form letter you can save it to disk, switch over to Extra Text (which you both enter and exit using Control X), and type in the sets of insertion data for each letter. To print out five versions of a form letter with 4 variable insertion blocks, you would have to enter 20 bits of information to Extra Text. To print out these 5 letters one after the other you use the command T (Type) L (List) C (Continuous) within the output mode. Incidentally, the action on the screen at this point, as the variable data blocks are dynamically inserted into your form letter, is one of the most impressive displays I've ever seen on the Pet!

One last word. This article was created using WordPro II - straight from my head to the screen of the Pet to one of our 3022 tractor drive printers.

NEW RELEASES (AVAILABLE MID JANUARY)

FOR THE COMMODORE LIBRARY

Educational

Introduction to Algebra	MP061	$20
Mathematical Games	MP062	$10
Sampling	MP063	$10
Languages	MP064	$10

Entertainment

Treasure Trove of Games 9	MP065	$10
Treasure Trove of Games 10	MP066	$10
Invaders	MP067	$07

Scientific/Mathematical/Engineering

R-L-C Circuit Analysis	MP070	$10
Drawing Load & Die Design	MP071	$10

We hope shortly to have both a tape and disk collection of utility programs available for general release. Although, this will probably be completed by the time you receive your newsletter. I don't want to officially sanction it's release yet, as we may be introducing some late changes to the package. We want to make sure you get the best!

Now a little more about the individual programs outlined above. Full descriptions will be published in our New Master Library Catalogue (to be published at the end of January). Make sure you get one from your local dealer.

MP061, AN INTRODUCTION TO ALGEBRA, is the first of what will be a complete suite of programs introducing the first time user to Algebra and numbers in general. Complete with manual, the 30 programs assume no prior knowledge or familiarity with Algebra, and step by step, guides the student through at his or her own pace.

Informally written and presented, people of any age should have no problem with coping with this excellent package, whether it be used at home or in the classroom.

Watch this space for future additions to our educational range.

Following on from this, MP063, SAMPLING, is an excellently written program,using the PET's graphics superbly, which gives very good demonstratins of what can be achieved with a knowledge of mathematics. As the title suggests, the progam covers statistical sampling and whether you want binomial, or Poisson or any of 6 kinds of distributions, with your own parameters, this program provides it.

MP064, LANGUAGES, allows you to brush up on your knowledge of French, German, Spanish, Italian, Dutch or Danish, all on one tape. Commodore enters the E.E.C. and at only $12.00 too! (You can't order in deutschmarks, francs, etc!).

On the entertainment side, MP065, TREASURE TROVE OF GAMES No 9, contains the following 4 programs:

1/ DODGE CITY - As sheriff of a wild west town, you discover ten outlaws on the loose! Can you shoot them, before you're sent on a one way trip to the cemetry?

2/ MOLECULES & ATOMS - PET version of the board game Black Box. Impesonate early 20th Century physicists as you try to discover the atomic structure of various molecules.

3/ HORSE RACE - Gamble on the horses without losing any money!

4/ ONE ARM BANDIT - Further attempts to keep out of the pubs and clubs of the land, allowing you to gamble for nothing! Very good graphics employed on this simulation of a popular pastime.

MP066 TREASURE TROVE OF GAMES No 10 contains -

1/ SQUADRON SCRAMBLE - Re-enact the Battle of Britain in the comfort of your home (or science laboratory) with the outcome at the start as uncertain as was the real thing. Excellent graphics.

2/ TOWER OF HANOI - Entertaining and complicated thinking involved in this ancient mathematical problem.

3/ SUB KILLER - Try to destroy as many enemy submarines as possile from your fully controlled ship.

MP067, INVADERS shows just what can be achieved in machine code game programming, graphics and real time simulation being surely used to the ultimate. (But we know you'll prove us wrong one day!). Complete with sound (wiring diagram displayed by program!), this must be the most addictive game Commodore has ever released. Many of you will have spent a fortune on this game in pubs and clubs by now. Buy from Commodore and save money!"

Incidentally, a copy of this program fell into the hands of Paul Higginbottom, one of our Software experts. His thoughts, well read on .....

You may not have heard of me before, my name is Paul Higginbottom and I am the latest addition to the Commodore Software Department, although I have been with Commodore for 6 months already in other capacities.

I deal mainly with the new Disk based packages but INVADERS was one tape program which I fell in love with.

This is written both entirely in machine code making it a graphically very attractive game. Anyway, now let's try to explain a little about the game:-

SPACE INVADERS - For those of you who have played the game in amusement arcades or public houses, then I would like to add that this is an exact copy of the game. Simply because the microprocessor in those arcade machines is a 6502, the very same as in the PET. For those of you who have not played it, the description may sound very silly, but those who have will tell you (me included) just how viciously addictive

the game is. In the game, you must move a space cannon undrneath your four bases and fire up at the fast approaching Space Invaders, which look like small sea monsters. They are confisured in rows across the screen and march across the screen. Each time any of the Space Invaders reaches the edge of the screen, they drop down one position, getting nearer and nearer to your bases. Just to add to your dilemma, they can fire at you too, droppins deadly dollar signs on you. You score by destroying the Invaders (the uppermost ones on the screen scorins highest). Also there is a mystery score which is a larger spaceship, which wanders across the top of your screen over the confusion below itselt. The Invaders score between 10 and 50 and the mystery is between 50 and 300. If you suceed in clearins all the Invaders off the screen, the screen simply resets another set of Invaders, but they are now a row nearer to you. End of game is caused by losins all three of your lives (4 if you score over 1500 and set the bonus life) or beins 'invaded' i.e. They land on earth. Movement is easy; holdins your finger on the 4 or 6 keys moves the sun left and right across the screen and pressins the A key fires the cannon. (Made possible because the PET keyboard is scanned and not interrupt driven like Apple). The PET plays itself in a demo, whi le you're not playing, sivins humorous instructions between each one. I would say that PET Space Invaders is slightly harder than the arcade version, but the cannon is more responsive to movement.

Other January releases include -

R-L-C CIRCUIT ANALYSIS, MP070, is exactly what the title describes, and goes hand in hand with our current package of Linear Circuit Analysis. The two between them contain much valuable informatin for anyone designins or usins electrical circuits of many kinds.

DIE DESIGN AND DRAWING LOAD, MP071, is a well written program whose function is again self evident from the title. In resonse to your inputs, the program will tell you whether the values that you have typed make sense or not. Probably my favourite of all the new releases.

On the business and financial side, you may feel that cassettes have been somewhat superceded by disks. Not everyone has a disk yet .... ! Commodore cassette business Software will continue to expand and improve to meet your srowins Software requirements and the New Year will continue to see that side of the Commodore Software library beins added to.

Mathematical, Scientific and Enginееrins is always a popular field for both programs and programmers, and a field that is particularly suited to the PET because of its' fine sаphics. We are always interested in receivins programs from new as well as established authors in this field.

Well here they are, the new cassette releases for the new year, as this year comes to a very interestins close. 1979 has seen many new releases, and our library now contains over 50 packages, all of which will run on the old and new ROMs. (Witness our release of Super 9*9 on MP044, as the last of the old ROM programs to be converted.) Our thanks and congratulations go out to all our authors for providins an excellent set of programs, which were a real pleasure to see throush to actual release. Keep it up!

Peter Gerrard
- Cassette Library Manager -

New/Old ROM Memory Map

SOME PET ROUTINES

By Jim Butterfield
and Jim Russell, Toronto

New		Old	
C2AA		C2AC-C2D9	peeks at the stack for an active FOR loop
C2D8		C2DA-C31C	"opens up' a space in Basic for insertion of a new line
C31B		C31D-C329	tests for stack-too-deep and aborts if found
C328		C32A-C356	check available memory space
C355		C357-C388	sends a canned error message from C190 area, then drops into:
		C389-C391	Signals 'ready'
C392	*	C394-C3A9	gets a line of input, analyzes it, executes it
C3AB	*	C3AC-C42E	handles a new line of Basic from keyboard; deletes old line, etc.
C439	*	C430-C460	corrects the chaining between Basic lines after insert/delete
C46F		C462-C476	receives a line from the keyboard into the Basic buffer
C481		C479-C48C	gets each character from keyboard
C495		C48D-C521	looks up the keywords in an input line and changes to "tokens"
C52C		C522-C550	searches for the location of a Basic line from number in 8,9
C55B		C551-C599	implements NEW command - clears everything
C5A7		C59A-C5A7	sets the Basic pointer to start-of-program
C5B5		C5A8-C647	performs LIST compand
C658		C649-C68F	executes a FOR statement
C6A1		C692-C6B4	continues to build FOR vectors
C6C4		C6B5-C6EF	reads and executes the next Basic statement, finds next line, etc.
C700	*	C6F2-C70A	executes the Basic Command as a subroutine
C730		C70D-C71B	performs RESTORE
C73F		C71C-C742	handles STOP, END, and BREAK procedures
C76B		C745-C75E	performs CONT
		C75F-C76D	set pause after carriage return (never called)
C577		C770-C772	performs CLR
C785		C775-C77D	performs RUN
C790		C780-C79A	performs GOSUB
C7AD		C79D-C7C9	performs GOTO
C7DA		C7CA-C7FD	performs RETURN
C80E		C7FE-C81E	scans for start of next Basic line
C830		C820-C840	performs IF
C853		C843-C862	performs ON
C873		C863-C89A	gets a fixed-point number from Basic and stores in 8,9
C8AD		C89D-C91B	performs LET
C928		C91C-C97E	checks numeric digit/move string pointer
C98B		C97F-C982	performs PRINT#
C991		C985-C996	performs CMD
C9A5		C999-CA24	performs PRINT
CA1C		CA27-CA41	prints string from address in Y,A
CA45	*	CA44-CA76	prints a character
CA4F		CA77-CA9E	handles bad input data
CA7D		CA9F-CAC5	performs GET
CAA7		CAC6-CADF	performs INPUT#
CAC1		CAE0-CB14	performs INPUT
CAFA		CB17-CB21	prompts and receives the input
CB07		CB24-CC11	performs READ
CBFC		CC12-CC35	canned messages: EXTRA IGNORED; REDO FROM START
CC20		CC36-CC8F	performs NEXT

New		Old	
CC79		CC92-CCB5	checks Basic format, data type, flags TYPE MISMATCH
CC9F		CCB8-CD38	inputs and evaluates any expression (numeric or string)
CD21		CD3A-CD9C	pushes a partially-evaluated argument to the stack
CD84		CD9D-CDB9	evaluates a numeric, variable, or pi, or identifies other symbol
CDA3		CDBC-CDCO	value of pi in floating binary
CDA8		CDC1-CDE7	checks for special characters (+, -, ", .) at start of expression
CDCF		CDE8-CDF6	performs NOT function
CDDE		CDF7-CE04	checks for various functions
CDEC		CE05	evaluates expression within parentheses ()
CDF2		CE0B	checks for right parenthesis)
CDF5		CE0E	checks for left parenthesis (
CDF8		CE11-CE1B	checks for comma
CE03		CE1C-CE20	prints SYNTAX ERROR and exits
CE08		CE21-CE27	sets up function for future evaluation
CE0F		CE28-CE39	set up a variable name search
CE2A		CE3B-CE96	checks for special variables TI, TI$, and ST
CE89		CE97-CED5	identifies and sets up function references
CEC8		CED60CF05	perform the OR and AND functions
CEF8		CF06-CF6D	performs comparisons
CF60		CF6E-CF7A	sets up DIM execution
CF6D		CF7B-D00E	searches for a Basic variable
D001		D00F-D078	creates a new Basic variable
D069		D079-D087	logs Basic variable location
D078		D088-D098	is array pointer subroutine
D089		D099-D09C	is 32768 in floating binary
D08D		D09D-D0B8	is floating point-to-fixed conversion for signed values
D0AC		D0B9-D263	locates and/or creates arrays
D259		D264-D277	performs FRE function
D26D		D278-D284	converts fixed point-to-floating
D27A		D285-D28A	performs POS function
D280		D28B-D294	checks direct/indirect command, gives 'ILLEGAL DIRECT'
D28D		D295-D348	executes DEF statements and evaluation FNx
D33F		D349-D36A	performs STR$ function
D361		D36B-D3D1	scans and sets up string elements
D3CE		D3D2-D403	builds string vectors
D400		D404-D5C3	does 'garbage collection' - discards unwanted strings
D5C6		D5C4-D5D7	performs CHR$ function
D5DA	*	D5D8-D653	performs LEFT$, RIGHT$, MID$ functions
D656		D654-D662	performs LEN, gets string length
D665		D663-D672	performs ASC function
D675		D673-D684	gets a single-byte value from Basic
D687		D685-D6C3	evaluates VAL function
D6C6		D6C4-D6CF	gets two arguments (16-bit and 8-bit) from Basic
D6D2		D6D0-D6E5	checks argument is in range 0-65535
D6E8		D6E6-D701	performs PEEK and POKE
D710		D702-D71D	executes WAIT statement
D72C		D71E-D890	performs addition and subtraction
D8C8		D891-D8BE	contains floating-point constants
D8F6		D8BF-D8FC	performs LOG function
D934		D8FD-D95D	performs multiplication
D998		D95E-D988	loads secondary accumulator from memory ($B8 to $BD)
D9C3		D989-D9B3	test and adjust primary/secondary accumulators
D9EE		D9B4-D9E0	routines to multiply or divide by 10
DA1B		D9E1-DA73	performs division
DAAE		DA74-DA98	loads primary accumulator from memory ($B0-$B5)

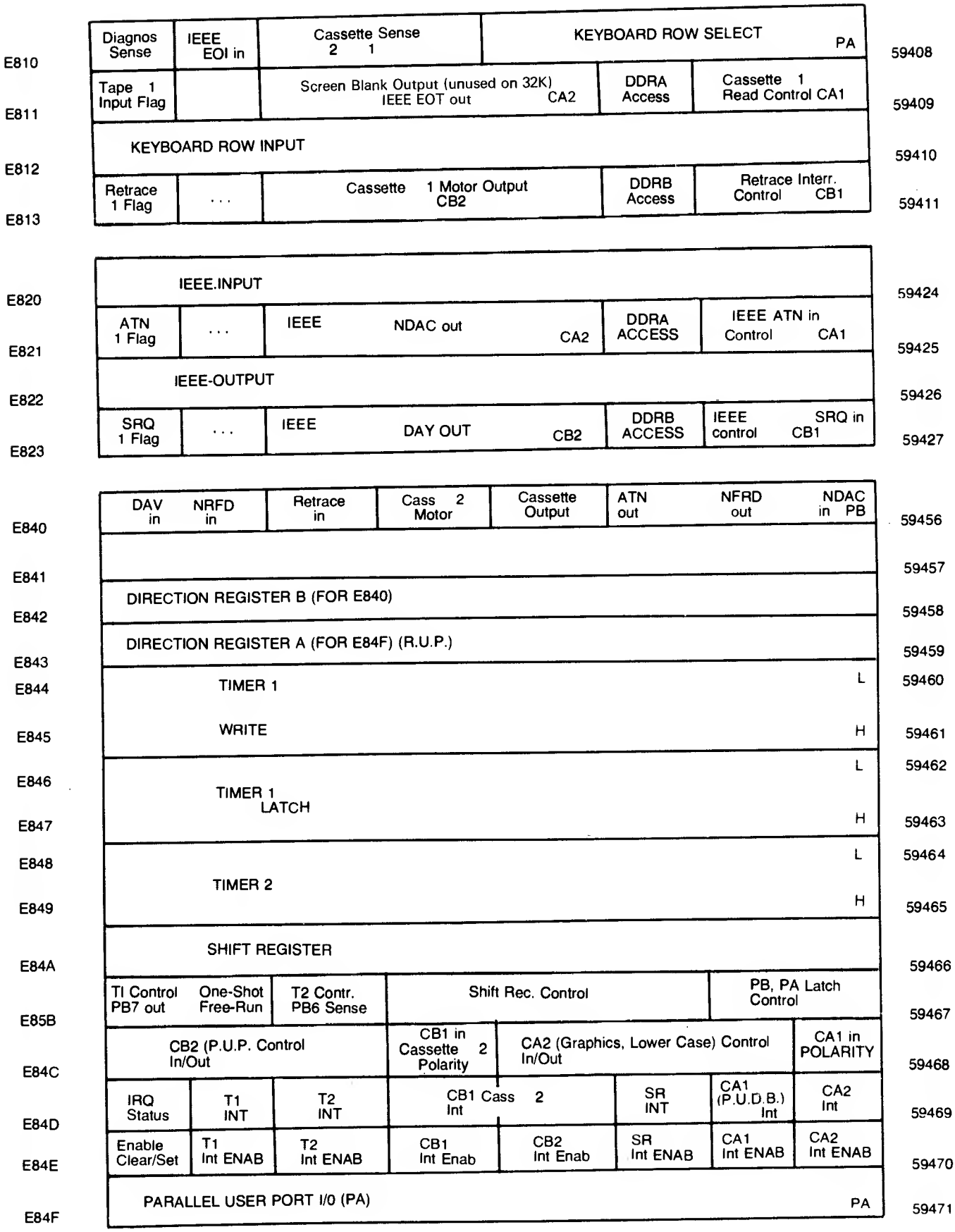
New		Old	
DAD3		DA99-DACD	transfers primary accumulator to memory
DB08		DACE-DADD	transfers secondary accumulator to primary
DB18		DADE-DAEC	transfers primary accumulator to secondary
DB27		DAED-DAFC	rounds the primary accumulator
DB37		DAFD-DB29	extracts primary sign; performs SGN function
DB64		DB2A-DB2C	performs ABS
DB67		DB2D-DB6C	compares primary accumulator to memory
DBA7		DB6D-DB9F	convert floating point to fixed, unsigned
DBD8		DB9E-DBC4	perform INT function
DBFF		DBC5-DC4F	convert ASCII string to floating point
DC8A		DC50-DC84	get new ASCII digit
DCCE		DC94-DCAE	print Basic line number
DCE9		DCAF-DDE2	convert floating point to ASCII string (at 0100 up)
DE1D		DDE3-DE23	conversion constants - decimal or clock
DE5E		DE24-DE2D	evaluation SQR function
DE68		DE2E-DE66	evaluation of power function
DEA1		DE67-DE71	negate (monadic -)
DEDA		DEA0-DEF2	perform EXP function
DF2D		DEF3-DF3C	perform function series evaluation
DF7F	*	DF45-DF9D	perform RND calculation
DFD8		DF9E	evaluate COS function
DFDF		DFA5-DEED	evaluate SIN function
E028		DFEE-E019	evaluate TAN function
E08C		E048-E077	evaluate ATN function
E0F9		E0B5-E0CC	Basic scan program, transferred to 00C2-00D9/0070-0087
E116		E0D2-E173	completion of power-on-reset; memory test, etc.
		E19B-E1BB	partial test for TI and TI$
		E1BC-E1E0	input/read/get director
E1DE	*	E1E1-E27C	initializes I/O registers, clear screen, reset subroutine
E285		E27D-E3C3	receive input from keyboard/screen
E3B4		E3C4-E3E9	set up new screen line
E3D8		E3EA-E52F	output character to screen
E519		E530-E5DA	check for and perform screen scrolling
E257		E5DB-E66A	start new screen line
E61B		E66B-E67D	interrupt entry
E6E4		E67E-E683	interrupt return
E62E		E685-E73E	hardware interrupt routine: cursor flash, tape motor, keyboard
E6F8		E73F-E7AB	convert keyboard matrix to ASCII
E6EA		E7AC-E7B9	write-on-screen subroutine
F156		E7DB-E7EB	print canned monitor message
F0B6		F0B6-F1CB	IEEE-488 channel open, test, close
F1D1		F1CC-F22F	get input character from keyboard, screen, cassette, IEEE
F232		F230-F27C	output character to screen, cassette, IEEE
F272		F27D-F2A3	restore normal I/O, clear IEEE channels
F26E		F2A4-F2AA	abort (not close!) all files
F28D		F2AB-F2B7	locate logical file table entry
F299		F2B8-F2C7	transfer file table entries to Device, Command
F2A9		F2C8-F329	perform file CLOSE
F301		F32A-F33E	test stop key
		F33F-F345	test if direct/indirect command for suppressing file advice
F3C2		F346-F3FE	perform file LOAD
		F3FF-F421	print "SEARCHING .."
		F422-F432	print "LOADING .." or "VERIFYING"
		F433-F461	get parameters for LOAD and SAVE
		F462-F494	perform IEEE sequences for LOAD, SAVE, and OPEN
		F495-F4BA	search for specific tape header

New	Old	
F4B7	F4BB-F4D3	perform VERIFY
	F4D4-F529	get parameters for OPEN and CLOSE
F521	F52A-F5AD	perform OPEN
F5A6	F5AE-F5E2	search for any tape header
	F5E3-F5EC	clear tape buffer
F5DA	F5ED-F64C	write tape header
F63C	F64D-F666	get start & end addresses from tape header
DF7F	DF45	New for RND(0)
E054	E01A	floating constants
E0BC	E078	floating constants
E116	E0D2	new initial SP value
E116	E0D2	minor differences in initialization
E1B7	E174	Bytes free, Commodore Basic
	E19B	Save line #, Print "READY" (see C751 K72B)
	E19F	Part of TI (see CE2A/CE33)
	E1AB	Part of TI (see CE2A/CE33)
	E1BC	Part of INPUT, GET, READ, etc. (see CB07/CB24)
	E1D9	Part of INPUT, GET, READ, etc. (see CB07/CB24)
	E1C2	?
	E1CC	?
E1DE	E1E1	initialization, (minor differences)
E246	E236	clear screen
E229	E250	initialize line ptrs for clear screen (minor changes)
E257	E5DB	adjust line ptrs for preset line
E285	E27D	get clear from KB buffer
E29D	E297	Wait for KB input, write to screen, exit on CR
E2F4	E2FA	INPUT from screen
E5BA	E605	
	E73F	
E6F8	E75C	
	E7AC	
E748	E7BC	
E761	E7D5	
	E7DE	
E76A		WROA
E775		WROB
E784		WRTWO
E78D		ASCII
E797		T2T2
E7A7		RDOA
E7B6		RDOB
E7E0		HEXIT
E7EB		RDOC
E7F7		ERROPR
F656	F667-F67C	Set buffer start address
F66C	F67D-F694	set tape buffer start and end pointers
F684	F695-F69D	perform SYS command
F69E	F69E-F71B	perform SAVE
	F71C-F735	find unused secondary address
F729	F736-F78A	update clock
F770	F78B-F7DB	set input device
F7BC	F7DC-F82C	set output device
F806	F82D-F83A	bump tape buffer counter
F812	F83B-F85D	wait for cassette PLAY switch
F835	F85E-F870	test cassette switch line

Monitor

New	Old	
F847	F871-F87E	wait for cassette RECORD and PLAY switches
F855	F87F-F8B8	read tape initiation routine
F886	F8B9-F8D1	write tape initiation routine
F89E	F8D2-F912	complete tape read or write
F8E6	F913-F91D	wait for I/O completion
F8F0	F91E-F92D	test stop key and abort if necessary
F900	F92E-F95E	subroutine to set tape read timing
F931	F95F-FBFB	interrupt routine for tape read
	FBDC-FBE4	save memory pointer
	FBE5-FBEB	set ST error flag
	FBEC-FBFF	subroutine to count 8 serial bits per byte
	FC00-FC1B	subroutine to write a bit to tape
	FC1C-FCFA	interrupt 1 for tape write - entry at FC21
	FCFB-FD15	terminate I/O and restore normal vectors
	FD16-FD37	subroutine to set interrupt vector
	FD38-FD47	power-on reset entry; test for diagnostic
	FD48-FD7B	diagnostic routine
	FD7C-FD8F	check sum routine
	FD90-FD9A	pointer advance subroutine
	FD9B-FFB1	diagnostic routines
		JUMP TABLE:
	FFC0	OPEN
	FFC3	CLOSE
	FFC6	set input device
	FFC9	set output device
	FFCC	restore normal I/O devices
	FFCF	input character (from screen)
	FFD2	output character
	FFD5	LOAD
	FFD8	SAVE
	FFDB	VERIFY
	FFDE	SYS
	FFE1	test stop key
	FFE4	get character from keyboard buffer
	FFE7	abort all I/O channels
	FFEA	update clock
	FFED-EFFA	turn off cassette motors
	FFFA-FFFB	NMI vector (mangled)
	FFFC-FFFD	reset vector
	FFFE-FFFF	interrupt vector

* = coding change


E810
E811
E812
E813
Diagnos Sense
IEEE EOI in
Cassette Sense 2 1
KEYBOARD ROW SELECT PA
59408
Tape 1 Input Flag
Screen Blank Output (unused on 32K) IEEE EOT out CA2
DDRA Access
Cassette 1 Read Control CA1
59409
KEYBOARD ROW INPUT
59410
Retrace 1 Flag
. . .
Cassette 1 Motor Output CB2
DDRB Access
Retrace Interr. Control CB1
59411
E820
E821
E822
E823
IEEE.INPUT
59424
ATN 1 Flag
. . .
IEEE NDAC out CA2
DDRA ACCESS
IEEE ATN in Control CA1
59425
IEEE-OUTPUT
59426
SRQ 1 Flag
. . .
IEEE DAY OUT CB2
DDRB ACCESS
IEEE SRQ in control CB1
59427
E840
E841
E842
E843
E844
E845
E846
E847
E848
E849
E84A
E85B
E84C
E84D
E84E
E84F
DAV in
NRFD in
Retrace in
Cass 2 Motor
Cassette Output
ATN out
NFRD out
NDAC in PB
59456
59457
DIRECTION REGISTER B (FOR E840)
59458
DIRECTION REGISTER A (FOR E84F) (R.U.P.)
59459
TIMER 1
L
59460
WRITE
H
59461
L
59462
TIMER 1 LATCH
H
59463
L
59464
TIMER 2
H
59465
SHIFT REGISTER
59466
TI Control PB7 out
One-Shot Free-Run
T2 Contr. PB6 Sense
Shift Rec. Control
PB, PA Latch Control
59467
CB2 (P.U.P. Control In/Out
CB1 in Cassette 2 Polarity
CA2 (Graphics, Lower Case) Control In/Out
CA1 in POLARITY
59468
IRQ Status
T1 INT
T2 INT
CB1 Cass 2 Int
SR INT
CA1 (P.U.D.B.) Int
CA2 Int
59469
Enable Clear/Set
T1 Int ENAB
T2 Int ENAB
CB1 Int Enab
CB2 Int Enab
SR Int ENAB
CA1 Int ENAB
CA2 Int ENAB
59470
PARALLEL USER PORT I/0 (PA)
PA
59471

To indentify a function of PET's original ROM, and/or convert it to the equivalent upgrade ROM location, use this table.

All addresses are given in hexadecimal.

OLD ADDRS	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0000:	0000	0001	0002	000E	**	**	**	**
0008:	0011	0012	0200	0201	0202	0203	0204	0205
0010:	0206	0207	0208	0209	020A	020B	020C	020D
0018:	020E	020F	0210	0211	0212	0213	0214	0215
0020:	0216	0217	0218	0219	021A	021B	021C	021D
0028:	021E	021F	0220	0221	0222	0223	0224	0225
0030:	0226	0227	0228	0229	022A	022B	022C	022D
0038:	022E	022F	0230	0231	0232	0233	0234	0235
0040:	0236	0237	0238	0239	023A	023B	023C	023D
0048:	023E	023F	0240	0241	0242	0243	0244	0245
0050:	0246	0247	0248	0249	024A	024B	024C	024D
0058:	024E	024F	0003	0004	0005	0006	0007	0008
0060:	0009	000A	000B	000C	000D	0013	0014	0015
0068:	0016	0017	0018	0019	001A	001B	001C	001D
0070:	001E	001F	0020	0021	0022	0023	0024	0025
0078:	0026	0027	0028	0029	002A	002B	002C	002D
0080:	002E	002F	0030	0031	0032	0033	0034	0035
0088:	0036	0037	0038	0039	003A	003B	003C	003D
0090:	003E	003F	0040	0041	0042	0043	0044	0045
0098:	0046	0047	0048	0049	004A	004B	004C	004D
00A0:	004E	004F	0050	0051	0052	0053	0054	0055
00A8:	0056	0057	0058	0059	005A	005B	005C	005D
00B0:	005E	005F	0060	0061	0062	0063	0064	0065
00B8:	0066	0067	0068	0069	006A	006B	006C	006D
00C0:	006E	006F	0070	0071	0072	0073	0074	0075
00C8:	0076	0077	0078	0079	007A	007B	007C	007D
00D0:	007E	007F	0080	0081	0082	0083	0084	0085
00D8:	0086	0087	0088	0089	008A	008B	008C	**
00E0:	00C4	00C5	00C6	00C7	00C8	00C9	00CA	00CB
00E8:	00CC	00B4	00CD	00CE	00CF	00D0	00D1	00D2
00F0:	00D3	00D4	00D5	00D6	00D7	00D8	00D9	00FB
00F8:	00FC	00DA	00DB	00DC	00DD	00DE	00DF	**
0200:	008D	008E	008F	0097	0098	0099	009A	00F9
0208:	00FA	009B	009C	009D	0096	009E	009F	026F
0210:	0270	0271	0272	0273	0274	0275	0276	0277
0218:	0278	0090	0091	0092	0093	00A0	00A1	**
0220:	00A3	00A4	00A5	00A6	00A7	00A8	00A9	00AA
0228:	00AB	00E0	00E1	00E2	00E3	00E4	00E5	00E6
0230:	00E7	00E8	00E9	00EA	00EB	00EC	00ED	00EE
0238:	00EF	00F0	00F1	00F2	00F3	00F4	00F5	00F6
0240:	00F7	00F8	0251	0252	0253	.. etc.		
0260:	00AC	00AD	00AE	00AF	00B0	00B1	00B2	**
0268:	00B5	**	**	**	00B7	**	**	00B9
0270:	00BA	00BB	00BC	00BD	00BE	00BF	00C0	00C1

Upper/Lower Case Converter

The following machine language program can be used to convert old ROM upper/lower case to the convention used by new ROM PETs.

LOAD and RUN the program below. Then LOAD the program you wish to convert for operation with new ROMs. Type SYS826 and well!... all upper case becomes lower case and, of course, vice versa. SAVE the converted program on tape or disk. Now you can LOAD and RUN the converted program in the usual manner.

```
100 FOR J=826 TO 925:READ A
110 POKE J,A:NEXT
130 DATA 169,4,133,202,169,1,133,,201
140 DATA 32,89,3,160,0,196,202,240,13
150 DATA 177,201,170,200,177,201,134
160 DATA 201,133,202,76,66,3,96,160,4
170 DATA 177,201,240,44,201,34,240,4
180 DATA 200,76,91,3,200,177,201,240
190 DATA 31,201,34,240,23,201,65,144
200 DATA 243,201,91,144,3,201,192,144
210 DATA 235,201,219,176,231,73,128
220 DATA 145,201,76,103,3,200,76,91,3
230 DATA 96,255,255,255,255,255,255
240 DATA 255,255,255,255,255,255,255
250 DATA 255,255,255,255,255
```

Trace

– Brett Butler, Toronto –

Yet another of the superb programs brought over the Atlantic by Jim Butterfield on the occassion of his last visit to England, this Trace program by Brett Butler is a very useful tool when debugging Basic programs. The whole of the line currently being executed (not just the line number!) is displayed in a reverse field "window" at the top of PETs screen.

The rate at which your program executes during debugging with TRACE, can be fine tuned with a SYS command, and SYS commands are available for switching TRACE on and off. When TRACE is enabled, your program can be run at approximately one quarter normal speed by simply depressing the "SHIFT" key.

This incerdible machine code program works equally well in both old and new ROM PETs, and will locate itself in whatever sized memory is fitted. It will also sit happily together with any other machine code utilities that you may have loaded into your PET such as SUPERMON, locating itself so that the facilities of all utilities are available concurrently!

The "intelligence" built into TRACE's locating routine gives rise to an unexpected side effect. If you load and run the Basic loader for Trace the program will give you the SYS commands appropriate for your paticular PET. If you ignore the instruction to note these down somewhere, you may need to load and run the Basic loader again to remind you of them. If you do this a number of times you will eventually notice that the SYS commands are differerent each time! A quick memory check - ?FRE(M) - every time you run the Basic Loader shows available RAM decreasing rapidly! What is happening? The Basic loader is locating TRACE alongside its previous copy in RAM. You have a number of copies of TRACE inside the machine each of which can be activated by the SYS commands appropriate to its particular memory location! If you need the "missing" RAM switch off and start again!

Our thanks to Brett Butler for his permission to reproduce TRACE in CPUCN.

```
50 PRINT"■■■THIS PROGRAM LOCATES ■TRACE■ IN"
60 PRINT"ANY SIZE MEMORY THAT IS FITTED...■"
65 IFPEEK(65E3)=254THEND=2:E=52:GOTO70
66 IFPEEK(65E3)<>192THENPRINT"?? I DON'T KNOW YOUR ROM ??":END
67 D=1:E=134:FORJ=1TO1E3:READX:IFX<1E4THENNEXTJ
70 PRINT"I SEE YOU HAVE AN ";
71 IFD=1THENPRINT"ORIGINAL";
72 IFD=2THENPRINT"UPGRADE";
73 PRINT"  R O M."
98 DATA -342,162,5,189,249,224,149,112,202,16,248,169,239,133,129,96
99 DATA 173,-342,133,52,173,-341,133,53,169,255,133,42,160,0,162,3
100 DATA 134,43,162,3,32,-271,208,249,202,208,248,32,-271,32,-271,76
101 DATA 121,197,162,5,189,-6,149,112,202,16,248,169,242,133,128,96
102 DATA 230,42,208,2,230,43,177,42,96,230,119,208,2,230,120,96
103 DATA 32,115,0,8,72,133,195,138,72,152,72,166,55,165,54,197
104 DATA 253,208,4,228,254,240,106,133,253,133,35,134,254,134,36,165
120 DATA 152,208,14,169,3,133,107,202,208,253,136,208,250,198,107,208
136 DATA 246,32,-54,169,160,160,80,153,255,127,136,208,250,132,182,132
153 DATA 37,132,38,132,39,120,248,160,15,6,35,38,36,162,253,181
169 DATA 40,117,40,149,40,232,48,247,136,16,238,216,88,162,2,169
185 DATA 48,133,103,134,102,181,37,72,74,74,74,74,32,-44,104,41
202 DATA 15,32,-44,166,102,202,16,233,32,-38,32,-38,165,184,197,119
221 DATA 240,55,165,195,208,4,133,253,240,47,16,42,201,255,208,8
237 DATA 169,105,32,-30,24,144,33,41,127,170,160,0,185,145,192,48
254 DATA 3,200,208,248,200,202,16,244,185,145,192,48,6,32,-32,200
271 DATA 208,245,41,127,32,-32,165,119,133,184,104,168,104,170,104,40
288 DATA 96,168,173,64,232,41,32,208,249,152,96,9,48,197,103,208
304 DATA 4,169,32,208,2,198,103,41,63,9,128,132,106,32,-54,164,182
322 DATA 153,0,128,192,195,208,2,160,7,200,132,182,164,106,96,76
333 DATA -255,32,-262
700 DATA 1E10
800 DATA-343,162,5,189,181,224,149,194,202,16,248,169,239,133,210,96
810 DATA173,-343,133,134,173,-342,133,135,169,255,133,124,160,0,162
820 DATA3,134,125,162,3,32,-272,208,249,202,208,248,32,-272,32,-272
830 DATA76,106,197,162,5,189,-6,149,194,202,16,248,169,242,133,210,96
840 DATA230,124,208,2,230,125,177,124,96,230,201,208,2,230,202,96,32
850 DATA197,0,8,72,133,79,138,72,152,72,166,137,165,136,197,77,208,4
860 DATA228,78,240,107,133,77,133,82,134,78,134,83,173,4,2,208,14,169
870 DATA3,133,74,202,208,253,136,208,250,198,74,16,246,32,-54,169,160
```

```
880 DATA160,80,153,255,127,136,208,250,132,76,132,84,132,85,132,86,120
890 DATA248,160,15,6,82,38,83,162,253,181,87,117,87,149,87,232,48,247
900 DATA248,160,15,6,82,38,83,162,2,169,48,133,89,134,88,181,84,72,74,74
910 DATA136,16,238,216,88,162,2,169,48,133,89,134,88,181,84,72,74,74
920 DATA74,74,32,-44,104,41,15,32,-44,166,88,202,16,233,32,-38,32,-38
930 DATA165,75,197,201,240,55,165,79,208,4,133,77,240,47,16,42,201,255
940 DATA208,8,169,94,32,-30,24,144,33,41,127,170,160,0,185,145,192,48
950 DATA3,200,208,248,200,202,16,244,185,145,192,48,6,32,-32,200,208
960 DATA245,41,127,32,-32,165,201,133,75,104,168,104,170,104,40,96,168
970 DATA173,64,232,41,32,208,249,152,96,9,48,197,89,208,4,169,32,208
980 DATA2,198,89,41,63,9,128,132,81,32,-54,164,76,153,0,128,192,79,208
990 DATA2,160,7,200,132,76,164,81,96,76,-256,32,-263
1000 S2=PEEK(E)+PEEK(E+1)*256:S1=S2+D-344
1010 FORJ=S1TOS2-1
1020 READX:IFX>=0GOTO1050
1030 Y=X+S2:X=INT(Y/256):Z=Y-X*256
1040 POKEJ,Z:J=J+1
1050 POKEJ,X
1060 NEXTJ
1070 PRINT"■■ === TRACE ==="
1080 REMARK: BY BRETT BUTLER, TORONTO
1090 PRINT"■TO INITIALIZE AFTER LOAD: SYS";S1+17
1100 PRINT"TO ENABLE TRACE:      SYS";S1+56 .
1110 PRINT"TO DISABLE:      SYS";S1+2
1120 PRINT"■CHANGE SPEED WITH:  POKE";S1+125-D;"■■,X"
1130 PRINT"■==MAKE A NOTE OF ABOVE COMMANDS=="
1140 PRINT"■SAVE USING MACHINE LANGUAGE MONITOR:"
1150 PRINT"   .S ";
1160 S=INT(S1/256):T=S1-S*256
1170 POKEE,T:POKEE+1,S
1180 POKEE-4,T:POKEE-3,S
1190 IFD=2THENPRINTCHR$(34);"TRACE";CHR$(34);",01";
1195 IFD=1THENPRINT" 01,TRACE";
1200 S=S1:GOSUB1400
1210 S=S2:GOSUB1400
1220 PRINT:END
1400 PRINT",";:S=S/4096
1410 GOSUB1420
1420 GOSUB1430
1430 T=INT(S):IFT>9THENT=T+7
1440 PRINTCHR$(T+48);:S=(S-INT(S))*16:RETURN
READY.
```

Supermon

```
100 PRINT"■■■■ SUPERMON! "
110 PRINT"■     DISSASSEMBLER ■D■ BY WOZNIAK/BAUM
120 PRINT"       SINGLE STEP ■I■ BY JIM RUSSO
130 PRINT"MOST OTHER STUFF ■,CHAFT■ BY BILL SEILER
140 PRINT"■BLENDED & PUT IN RELOCATABLE FORM"
150 PRINT"    BY JIM BUTTERFIELD"
155 POKE42,182:POKE43,6:CLR
160 L=PEEK(52)+PEEK(53)*256
170 N=L-1466:P=3391:FORJ=L-1TONSTEP-1
180 X=PEEK(P):IFX>0GOTO190
185 P=P-2:X=PEEK(P+1)+PEEK(P)*256:IFX=0GOTO190
186 X=X+L-65536:X%=X/256:X=X-X%*256:POKEJ,X%:J=J-1
190 POKEJ,X:P=P-1:PRINT"■";X;"■■  ":NEXTJ
200 X%=N/256:Y=N-X%*256:POKE52,Y:POKE53,X%:POKE48,Y:POKE49,X%
210 PRINT"■■LINK TO MONITOR -- SYS";N
220 PRINT:PRINT"SAVE WITH MLM:"
230 PRINT".S ";CHR$(34);"SUPERMON";CHR$(34);",01":X=N/4096:GOSUB250
240 X=L/4096:GOSUB250:END
250 PRINT",";:FORJ=1TO4:X%=X:X=(X-X%)*16:IFX%>9THENX%=X%+7
260 PRINTCHR$(X%+48);:NEXTJ:RETURN
READY.
```

```
10 REM SUPERMON INSTR
1100 GOSUB10000
1200 PRINT"■■■ SIMPLE ASSEMBLER "
1300 PRINT"■. ■A■  ■2000■  ■LDA■  ■#$12
1310 PRINT".A 2002  ■STA■  ■$8000,X
1320 PRINT".A 2005  ■(RETURN)
1330 PRINT".
1340 PRINT"     IN THE ABOVE EXAMPLE THE USER
1350 PRINT"STARTED ASSEMBLY AT 1000 HEX.  THE
1360 PRINT"FIRST INSTRUCTION WAS LOAD A REGISTER
1370 PRINT"WITH IMMEDIATE 12 HEX.  IN THE SECOND
1380 PRINT"LINE THE USER DID NOT NEED TO TYPE THE
1390 PRINT"A AND ADDRESS.  THE SIMPLE ASSEMBLER
1400 PRINT"PROMPTS WITH THE NEXT ADDRESS.  TO EXIT
1410 PRINT"THE ASSEMBLER TYPE A RETURN AFTER THE
1420 PRINT"THE ADDRESS PROMPT.  SYNTAX IS THE SAME
1430 PRINT"AS THE DISASSEMBLER OUTPUT.
1450 GOSUB9000
1500 PRINT"■■■■■ CALCULATE BRANCH "
1510 PRINT"■. ■C■  ■1000■  ■1010■  0E
1520 PRINT"■     THE EXAMPLE CALCULATES THE SECOND
1530 PRINT"BYTE OF A BRANCH INSTRUCTION.  THE
1540 PRINT"BRANCH OP-CODE IS AT 1000 HEX AND THE
1550 PRINT"TARGET ADDRESS IS 1010 HEX.  ■SUPERMON
1560 PRINT"RESPONDED WITH THE 0E HEX OFFSET.
1570 GOSUB9000
1600 PRINT"■■■ DISASSEMBLER "
1610 PRINT"■. ■D■  ■2000■
1620 PRINT"■(SCREEN CLEARS)
1630 PRINT".;  2000 A9 12       LDA #$12
1640 PRINT".;  2002 9D 00 80    STA $8000,X
1650 PRINT".;  2005 AA          TAX
1660 PRINT".;  2006 AA          TAX
1670 PRINT"(FULL PAGE OF INSTRUCTIONS)
1700 PRINT"■     DISASSEMBLES 22 INSTRUCTIONS
1710 PRINT"STARTING AT 1000 HEX.  THE THREE BYTES
1720 PRINT"FOLLOWING THE ADDRESS MAY BE MODIFIED.
1730 PRINT"USE THE CRSR KEYS TO MOVE TO AND MODIFY
1740 PRINT"THE BYTES.  HIT RETURN AND THE BYTES
1750 PRINT"IN MEMORY WILL BE CHANGED.  ■SUPERMON■
1760 PRINT"WILL THEN DISASSEMBLE THAT PAGE AGAIN.
1790 GOSUB9000
1800 PRINT"■■■ SINGLE STEP "
1810 PRINT"■. ■I
1820 PRINT"■     ALLOWS A MACHINE LANGUAGE PROGRAM
1830 PRINT"TO BE RUN STEP BY STEP.
1840 PRINT"■CALL REGISTER DISPLAY WITH .■R■ AND SET
1850 PRINT"THE PC ADDRESS TO THE DESIRED FIRST
1860 PRINT"INSTRUCTION FOR SINGLE STEPPING.
1870 PRINT"THE .■I■ WILL CAUSE A SINGLE STEP TO
1880 PRINT"EXECUTE AND WILL DISASSEMBLE THE NEXT.
1890 PRINT"■CONTROLS:
1900 PRINT"   ■C■ FOR SINGLE STEP;
1910 PRINT"   ■RVS■ FOR SLOW STEP;
1920 PRINT"   ■SPACE■ FOR FAST STEPPING;
1930 PRINT"   ■STOP■ TO RETURN TO MONITOR."
1990 GOSUB9000
2000 PRINT"■■■■ FILL MEMORY "
2010 PRINT"■■. ■F■  ■1000■  ■1100■  ■FF
2020 PRINT"■     FILLS THE MEMORY FROM 1000 HEX TO
2030 PRINT"1100 HEX WITH THE BYTE FF HEX.
2090 GOSUB9000
2100 PRINT"■■■■ GO RUN "
2110 PRINT"■. ■G
2120 PRINT"■     GO TO THE ADDRESS IN THE PC
2130 PRINT"REGISTER DISPLAY AND BEGIN RUN CODE.
2140 PRINT"ALL THE REGISTERS WILL BE REPLACED
2150 PRINT"WITH THE DISPLAYED VALUES."
2160 PRINT"■. ■G■  ■1000
2170 PRINT"■     GO TO ADDRESS 1000 HEX AND BEGIN
2180 PRINT"RUNNING CODE.
2190 GOSUB9000
2200 PRINT"■■■■ HUNT MEMORY "
2210 PRINT"■. ■H■  ■C000■  ■D000■  ■'READ
2220 PRINT"■     HUNT THRU MEMORY FROM C000 HEX TO
2230 PRINT"D000 HEX FOR THE ASCII STRING ■READ■ AND
2240 PRINT"PRINT THE ADDRESS WHERE IT IS FOUND.  A
```

```
2250 PRINT"MAXIMUM OF 32 CHARACTERS MAY BE USED.
2260 PRINT"■.■H■ ■C000■ ■D000■ ■20■ ■D2■ ■FF
2270 PRINT"■     HUNT MEMORY FROM C000 HEX TO D000
2280 PRINT"HEX FOR THE SEQUENCE OF BYTES 20 D2 FF
2290 PRINT"AND PRINT THE ADDRESS.  A MAXIMUM OF 32
2300 PRINT"BYTES MAY BE USED.
2390 GOSUB9000
2400 PRINT"■■ LOAD FROM TAPE "
2401 PRINT"■■.■L■
2402 PRINT"■   LOAD ANY PROGRAM FROM CASSETTE #1.
2403 PRINT"■■.■L■ ■";CHR$(34);"RAM TEST";CHR$(34)
2404 PRINT"■   LOAD FROM CASSETTE #1 THE PROGRAM
2405 PRINT"NAMED ■RAM TEST■.
2410 PRINT"■■.■L■ ■";CHR$(34);"RAM TEST";CHR$(34);",02■
2420 PRINT"■   LOAD FROM CASSETTE #2 THE PROGRAM
2430 PRINT"NAMED ■RAM TEST■.
2490 GOSUB9000
2500 PRINT"■■■ MEMORY DISPLAY "
2510 PRINT"■.■M■ ■0000■ ■0080
2520 PRINT"■.:  0000 00 01 32 03 04 05 06 07
2530 PRINT".:  0008 08 09 0A 0B 0C 0D 0E 0F
2540 PRINT"■     DISPLAY MEMORY FROM 0000 HEX TO
2550 PRINT"0080 HEX.  THE BYTES FOLLOWING THE
2560 PRINT"ADDRESS MAY BE MODIFIED BY EDITING AND
2570 PRINT"THEN TYPING A RETURN.
2590 GOSUB9000
2600 PRINT"■■■ REGISTER DISPLAY "
2610 PRINT"■.■R
2620 PRINT"■     PC  IRQ  SR AC XR YR SP
2630 PRINT".;  0000 E52E 01 02 03 04 05
2640 PRINT"■     DISPLAYS THE REGISTER VALUES SAVED
2650 PRINT"WHEN ■SUPERMON■ WAS ENTERED.  THE VALUES
2660 PRINT"MAY BE CHANGED WITH THE EDIT FOLLOWED
2670 PRINT"BY A RETURN.
2671 PRINT"■  USE THIS INSTRUCTION TO SET UP THE
2672 PRINT"PC VALUE BEFORE SINGLE STEPPING WITH"
2673 PRINT".■I■
2690 GOSUB9000
2700 PRINT"■■■■ SAVE TO TAPE "
2710 PRINT"■.■S■ ■";CHR$(34);"PROGRAM NAME";CHR$(34);",01,0800,0C80"
2720 PRINT"■     SAVE TO CASSETTE #1 MEMORY FROM
2730 PRINT"0800 HEX UP TO BUT NOT INCLUDING 0C80
2740 PRINT"HEX AND NAME IT ■PROGRAM NAME■.
2790 GOSUB9000
2800 PRINT"■■■■■ TRANSFER MEMORY "
2810 PRINT"■.■T■ ■1000■ ■1100■ ■5000
2820 PRINT"■     TRANSFER MEMORY IN THE RANGE 1000
2830 PRINT"HEX TO 1100 HEX AND START STORING IT AT
2840 PRINT"ADDRESS 5000 HEX.
2890 GOSUB9000
3200 PRINT"■■■■ EXIT TO BASIC "
3210 PRINT"■.■X
3220 PRINT"■     RETURN TO BASIC READY MODE.
3230 PRINT"THE STACK VALUE SAVED WHEN ENTERED WILL
3240 PRINT"BE RESTORED.  CARE SHOULD BE TAKEN THAT
3250 PRINT"THIS VALUE IS THE SAME AS WHEN THE
3260 PRINT"MONITOR WAS ENTERED.  A CLR IN
3270 PRINT"BASIC WILL FIX ANY STACK PROBLEMS.
3290 GOSUB9000
3500 PRINT"■         SUMMARY                    "
3505 PRINT"COMMODORE MONITOR INSTRUCTIONS:"
3510 PRINT"■G■ GO RUN
3520 PRINT"■L■ LOAD FROM TAPE
3530 PRINT"■M■ MEMORY DISPLAY
3540 PRINT"■R■ REGISTER DISPLAY
3550 PRINT"■S■ SAVE TO TAPE
3560 PRINT"■X■ EXIT TO BASIC
3570 PRINT"■H■ HUNT MEMORY
3580 PRINT"■L■ LOAD FROM TAPE
3590 PRINT"■M■ MEMORY DISPLAY
3595 PRINT"■SUPERMON ADDITIONAL INSTRUCTIONS:"
3600 PRINT"■■A■ SIMPLE ASSEMBLER
3610 PRINT"■C■ CALCULATE BRANCH
3620 PRINT"■D■ DISASSEMBLER
3630 PRINT"■F■ FILL MEMORY
3640 PRINT"■H■ HUNT MEMORY
3650 PRINT"■I■ SINGLE STEP
```

```
3660 PRINT"T TRANSFER MEMORY
4000 GOSUB9000
4010 PRINT"SUPERMON WILL LOAD ITSELF INTO THE
4020 PRINT"TOP OF MEMORY .. WHEREVER THAT HAPPENS.
4030 PRINT"TO BE ON YOUR MACHINE.
4040 PRINT"   YOU MAY THEN SAVE THE MACHINE CODE
4050 PRINT"FOR FASTER LOADING IN THE FUTURE.
4060 PRINT"BE SURE TO NOTE THE SYS COMMAND WHICH
4070 PRINT"LINKS SUPERMON TO THE COMMODORE
4080 PRINT"MONITOR.
5000 GOSUB9000
8000 GOTO1200
9000 PRINT" HIT ANY KEY TO CONTINUE "
9010 FORI=1TO10:GETA$:NEXT
9020 GETA$:IFA$=""THEN9020
10000 PRINT"          S U P E R M O N 1 . 0 "
10010 PRINT"COMMANDS - USER INPUT IN REVERSE "
10040 RETURN
READY.
```

Programming

HOT TIPS

– Paul Hickinbotham –

First I would like to answer some frequently raised questions about screen formatting of data, and then take a look at a few techniques to make programs smaller and more elegant.

Formatting numbers on the screen can cause problems when the TAB function is used. If a number is to be printed within a box, then it would be nice to ensure that the last digit of the number always touches the right hand side of the box.

For example, if the number is simply TABbed into the box, and the number is a "0", then it will appear at the left hand side of the box, which doesn't look very smart. It is therefore necessary to use the length of the number (i.e the number of digits including decimal points) to drive the TAB expression. A number has a leading space and a trailing cursor right which needs to be taken into consideration. The LEN function counts the number of characters that there are in a string. In order to use LEN it is first necessary to convert the number into a string variable using STR$. The number of characters in the number is given by –

```
X=LEN(STR$(A))-1
```

– where A is the number. 1 is subtracted to take account of the leading space. So now, taking the above example, if we were to TAB(11-X) we would be in business!

Sometimes it is desirable to tack leading zeroes onto integer numbers (ie. to display "0038" rather than "38"), but this is a little more tricky to program.

Let the number be S. Let S$ be the string version of S with leading zeroes then

```
S$=RIGHT$("0000"+MID$(STR$(S),2),4)
```

MID$(STR$(S),2) converts S into a string without the leading space since it takes the string version of S from the 2nd character onwards. Then the four rightmost characters of the string of zeroes plus the shortened string version of S are concatenated to form S$.

* * *

Programs can be shortened a great deal with a little thought and an active imagination. For example it is often necessary to set a flag if a condition is met or to compliment the flag. The usual code for complimenting a flag is:-

```
1200 IFG=0THENG=1:GOTO1220
1210 G=0
1220 .....
```

On consideration, the statement –

```
1200G=1-G
1220 .....
```

– will be seen to have the same effect.

* * *

The two programs which follow are further examples of compact coding.

The first is a screen dump routine which makes it possible to copy the screen onto the printer at any time. For example a graphics display on the screen can be transformed into hard copy by means of a GOSUB to this routine.

When a screen dump is performed it is necessary to read the screen, and then turn it into a printable format. The easiest way to read the screen is to use the PEEK command, but the printer requires ASCII codes which are different from PEEK/POKE codes, and so a conversion is necessary. Now this can be done by a series of IF THEN statements but the result is rather horrible.

```
 5 REM CHARACTER SET
 6 PRINT"□"
 10 FORI=0TO255:POKE32768+I,I:NEXT
 15 PRINT"■■■■■■■"
 50 REM    * * *  QUICK AND DIRTY SCREEN PRINT  * * *
 100 OPEN4,4:PRINT#4,"":OPEN3,4,6
 102 PRINT"■NORMAL LINE FEED OR CLOSED UP? ■N■ OR ■C■":FORI=1TO10:GETA$:NEXT
 103 GETA$:IFA$=""THEN103
 104 N=18:IFA$="N"THENN=24
 105 PRINT#3,CHR$(N)
 110 FORI=0TO999
 130 P=PEEK(32768+I)
 134 GOSUB500
 135 IFP<64THENP=P+64:GOTO200
 140 IFP<126THENP=P+128:GOTO200
 145 IFP<128THENP=P+64:GOTO200
 150 IFP<191THENP=(P-64):GOTO200
 155 IFP=255THENP=191:GOTO200
 200 PRINT#4,CHR$(P);:
 220 X=X+1:IFX=40THENPRINT#4,"":X=0:F=0
 240 NEXT
 250 PRINT#4,"":CLOSE4
 255 PRINT#3,"":CLOSE3
 300 END
 500 REM REVERSE FLAG
 510 IFF=1ANDP>127THEN600
 515 IFP>127THENF=1:PRINT#4,"■";:GOTO600
 520 IFF=0ANDP<127THEN600
 530 IFP<127THENF=0:PRINT#4,"■";:GOTO600
 600 RETURN
READY.
```

```
NORMAL LINE FEED OR CLOSED UP? ■N OR ■C
```

Line 10 prints out PET's upper case and graphics character set onto the screen to provide some test characters. It should be omitted when the program is used in "real life". Line 15 can have the number of "cursor downs" adjusted so that the "Normal line feed... " prompt in line 102 does not overprint a crucial area of the screen. In a situation where the whole of the screen is important this part of line 102 should be removed altogether.

The main code conversion routine lies between lines 110 and 240; the subroutine 500-600 deals with reverse field characters; lines 100-105 allow the user to close-up the line feed (useful when printing graphics); lines 250, 255 ensure the printer buffer is emptied at the end of the program.

Now let's have a closer look at the differences between PEEK/POKE codes and ASCII. If one considers the binary representations of a number of different characters it will become apparent that only the 3 most significant bits (bits 5,6,7) are changed. "Aha, a bit of BOOLEAN ALGEBRA will solve this problem!" Using the OR and AND functions it is possible to convert PEEK/POKE codes to ASCII in just one statement.

Thus if A=PEEK(32768) (the top left hand corner of the screen), then -

```
    B=(AAND128)OR((AAND664)*2)OR((64-AA
NID32)*2)
```

- where B is the corresponding ASCII code.

The next matter to take care of is that the screen is 40 characters wide and the printer is 80, so it is necessary to check when 40 characters have been sent down to the printer and then to do a carriage return. This can be done by considering the screen as a thousand address locations between 1 and 1000, then looking at the particular screen

address and seeing if it is divisible by 40.

Here's the new program -

```
 10 OPEN4,4:PRINT#4," ____________
____________":PRINT#4,"¯";
 20 T=40:FORI=32768TO33767:A=PEEK(I):B=
(AANDI27)OR((AANDI64)*2)OR(64-AANDI32)*2)
 30 PRINT#4,CHR$(B);:IF(I-32807)/T=INT(
(I-32807)/T)THENPRINT#4,"I "CHR$(13)"¯";
 40 NEXT:PRINT#4,"_____________":CLOSE4
READY.
```

The lines are there to put a box around the display.

This routine will not output RVS (reverse field) characters as RVS but this can be fixed because RVS characters have a PEEK/POKE code greater than 127 and so AANDI28 will be 128 if the character is in RVS and 0 if not so if before each character we output CHR$(146-AANDI28) this will put out a RVS ON character if the character is in RVS otherwise an OFF RVS will be output to counteract any RVS ON's that had been printed. (Once the RVS has been switched on it stays on until the next carriage return.) The only bug I have observed with this fix is between quotes (the ON and OFF RVS symbols are printed literally), but this can be avoided by remembering the position on the line and doing a carriage return without line feed - CHR$(141) (which gets you out of 'quotes mode'), and returning to the same point on the line and continuing.

I hope that this whets your appetite for doing some exploration of your own. If you do, then please send your discoveries to the editor so that we can all share them via the medium of CPUCN.

My final program has no application apart from fun, but if there is anyone holding a raffle then this might be a good item in similar context to 'GUESS THE WEIGHT OF THE CAKE' ETC.. It is a PSEUDO ABACUS or bead counter:-

By using the symbols -------*------ we can make the bead appear to move along the line:-

```
1.--------*-------
2.---------*------
3.----------*-----
```

To make the initial display:-

```
▒*----------------▒
▒**---------------▒
▒***--------------▒
▒****-------------▒
▒*****------------▒
▒******-----------▒
▒*******----------▒
▒********---------▒
▒*********--------▒
▒**********-------▒
```

```
 10 PRINT"<clr>ABACUS:-":FORI=1TO9:PRIN
T"▒":FORJ=1TOI:PRINTCHR$(209);:NEXTJ
 20 FORJ=ITO22:PRINTCHR(192);:NEXTJ:PRI
NT"▒" :NEXTI
```

The next job is to create an array with the POKE addresses of the mid points of each row:-

```
 30 FORI=1TO9:P(I)=33179-I*40:NEXT
```

Using variables instead of constants gives a faster execution time -

```
 40 Q=12:D=9:E=102:L=64:B=81:R=13:X=14:
S=1
```

E = EDGE POKE CODE OR THE ▒ SYMBOL
B = BEAD CODE
S = STARTING ROW
L = LINE CODE (BAR OF ABACUS)

The routine uses a scanning technique that scans from the middle of the bottom row moving left until it finds something that is not a line. If it is a bead, then it pushes that bead along the bar until that bead hits something that is not the bar (either the right hand edge or another bead), but if it is the left hand edge then it knows that all the beads have been moved over to the right hand side and so it moves then back to the start and tries to move one bead of the next row up along using the same scanning technique. When a full scan has finished it returns to the bottom row and starts again (It really is difficult trying to explain this, but the idea is not that difficult.)

Here's the complete program listing -

```
 100 PRINT"<clr>ABACUS:-":Q=12:D=9
 105 E=102:L=64:B=81:R=13:X=14:S=1
 110 FORI=1TO9:PRINT"▒";:FORJ=1TOI
 120 PRINTCHR$(209);:NEXTJ
 130 FORJ=ITO22:PRINTCHR$(192);:NEXTJ:P
RINT"▒":NEXTI
 140 FORI=1TO9:P(I)=33179-I*40:NEXT
 150 C=S:REM"SETS ROW TO 1"
 160 P=P(C):REM"SET SCAN LINE POS'N"
 170 IFPEEK(P)=LTHENP=P-S:GOTO170
 180 IFPEEK(P)=ETHEN210
 190 POKEP,L:P=P+S:POKEP,B:IFPEEK(P+S)=
LTHEN190
 200 GOTO150
 210 T=D-C:FORI=STOX+C-S:POKEP(C)+R-I,L
:POKEP(C)+Q-I-T,B:NEXT:C=C+S:GOTO160
```

One thing I found is that the beads move very fast , if you wish to slow them down you can add in a delay loop in line 190 :-

```
 190 POKEP,L:P=P+S:POKEP,B:IFPEEK(P+S)=
LTHENFORI=1TO20:NEXT:GOTO190
```

I will leave you with a final puzzle - how long will it take before the top bead moves?

For/Next Loop Structures

- Jim Butterfield -

Recent remarks on popular Basic implementations indicate that difficulties may be encountered if the programmer jumps out of a FOR/NEXT loop.

This would be very serious if true. The programmer doing a table search would be required to continue scanning the table even after finding the item he wants; or to use questionable practices such as meddling with the loop variable while still within the loop.

Fortunately, it's true only for a few complex situations - and these are easy to fix if you understand how the dynamic FOR/NEXT loop works. (Dynamic loops are those set up during an actual program run, as contrasted to pre-compiled loops which are checked out before the actual run starts).

When a dynamic interpreter such as Microsoft Basic, encounters a statement such as FOR=J ... it sets up internal tables to manage the loop. These internal tables contain such things as: where to return if a NEXT J is encountered; the identity of the loop variable (in this case, J); whether the loop is counting up or down, etc.

These tables will remain until one of three things happens. If the loop goes through its complete range (by encountering a suitable number of NEXT J statements); if a new FOR J statement is found; or if a higher priority loop is terminated for either of the previous reasons.

The last rule is very sensible, and it's worth a closer look. Suppose we have set up a sequence of commands such as: FOR I= ... : FOR J= ... : For K= ... , and suppose the computer, while dealing with these three loops, finds a new FOR I= ... statement. It very wisely says, in its own computerese, "OK - looks like the big loop is being restarted; so the little ones are finished, too". And it promptly terminates the J and K loops, removing the tables from its memory.

Exactly what we want - but there are a couple of hidden "gotchas" that the user must know about when he gets into tricky coding routines.

The easiest one to spot is the situation where every loop has a different variable name. The first loop is, say, FOR A... the next one, FOR B... and the programmer continues through the alphabet with each loop. His idea is good: he can analyse how each loop has behaved, for each variable remains untouched for his examination. But each time he jumps out of a loop, the loop tables remain in memory, using up valuable stack or table space. He'd be much better off to give at least his outer loops the same variable name, and reclaim that space.

The second problem spot is a little more subtle, and an example would best illustrate it.

Here's a simple program to input a string, extract the individual words (eliminating single or multiple spaces), and print them:

```
100 INPUT S$      (get the string
110 K=1           (mark start-of-string
120 FOR J=K TO LEN(S$)
130 IF MID$(S$,J,1)<>""GOTO 150
                  (skip spaces
140 NEXT J
150 IF J>LEN(S$) GOTO 900
160 FOR K=J TO LEN(S$)
170 IF MID$(S$,K,1)=""GOTO 200
                  (scan to space or end
180 NEXT K
200 PRINT MIN$(S$,J,K-J)
800 IF K< =LEN(S$) GOTO 120
900 END
```

This program works quite well, and isn't hard to follow. It should be noted that if either the J or K loops run to completion, the variables will have a value of LEN(S$)+1; this is intended and allowed for in lines 150 and 800.

Before we extend this program into catastrophe, let's note one thing: by the time the program reaches line 200, both the J and K loops will still be open most of the time - we "jumped out" of both of them. No real problem; when we go back to 120, the new FOR J= .. will cancel them both.

Now let's get into trouble. We may be writing a little ELIZA here, and want to check the word we've found against a table of keywords so as to pick a suitable reply. We'll assume a table of twenty keywords, and start to build a search loop. Replacing line 200, we start a new loop:

```
200 X$ = MID(S$,J,K-J)     (set word
210 FOR I=1TO20
```

Our loop is now three deep - J and K are still considered active, remember? No problem with three-level loops; we're still OK.

But here's where we might get clever and wreck everything. We need to preserve K - that's where our search for the next word will start. But J has served its purpose and could be used again, right? Well... let's see.

This table of 20 words is really a double table. It contains pairs of words such as "I", "YOU", or "MY", "YOUR". To make our computer talk we must spot a word from either column, and switch in the word from the opposite column (so that "I HAVE FLEAS" will become "YOU HAVE FLEAS"). So we need one more loop to search over the two columns.

Let's be clever and use J, since we have decided that it isn't needed any more at this point. We code:

```
 220 FOR J=1 TO 2
 230 IF X$=T$(I,J) THEN X$=T$(I,3-J):GO
TO 400                          (swap word
 240 NEXT J
 250 NEXT I
 400 PRINT X$;"";               (repeat word
```

Suddenly everything stops working and the world tumbles down around our program. What happened?

Let's stop and analyse. Just before executing line 220, the computer had three active loops with variables J,K and I. Now it reaches line 220, and what does it see? A loop based on J, the "biggest" loop! So what does it do? It cancels the K and I loops, of course, and starts a new J loop.

When we reach line 250, the computer sees NEXT I - but it no longer has an active FOR I= loop, and you get a NEXT WITHOUT FOR error message.

The rule here is slightly more complex, but not too tough. If you use J as an "outer" loop variable, never use it for an inner loop. If we reversed I and J in the coding from 210 to 250, we'd have no problem. Try to think in terms of the hierarchy of loops, and you can make sure that a given variable is used only at its proper hierarchal level.

Let's try to put the rules together and create a tiny ELIZA, polishing up some of the coding as we go. You'll have fun adding your own features to it.

```
 100 DIM T$(1,4)      (two by five array
 110 DATA ME,YOU,I,YOU,MY,YOUR,AM,ARE,M
YSELF,YOURSELF
 120 FOR J=0 TO 4
 130 FOR K=0 TO 1
 140 READ T$(J,K)
 150 NEXT K
 160 NEXT J
 170 INPUT S$
 180 K1=1
 190 FOR J=K1TOLEN(S$)
 200 IF MID$(S$,J,1)=""THEN NEXT J
 210 J1=1
 220 IF J>LEN(S$) GOTO 900
 230 FOR J=J1 TO LEN(S$)
 240 IF MID$(S$,J,1)<>""THEN NEXT J
 250 K1=J
 260 X$=MID$(S$,J1,K1-J1)
 270 FOR J=0 TO 4
 280 FOR K=0 TO 1
 290 IF T$(K,J)=X$ THEN X$=T$(1-K,J):GO
OTO 320
 300 NEXT K
 310 NEXT J
 320 PRINT"";X$;
 330 IF K1<=LEN(S$)GOTO 190
 340 PRINT "?"
 900 GOTO 170
```

Note that the outermost loop is now always called J, the next down always K. I've tightened up the array to use the zero rows and columns to save memory, and the search loops are a little faster.

Even though the program is riddled with premature loop exits, there are no problems. Just observe a few simple rules and you will have efficient and trouble-free loops.

PETSOFT PROGRAMMERS TOOLKIT

"10 POWERFUL NEW COMMANDS FOR YOUR PET!"

The Toolkit is a machine language program which is provided in a 2 kilobyte ROM chip. Just plug it in — no tools are necessary — and your PET's BASIC has 10 new and very useful commands:

AUTO	Provides new line numbers when you are entering BASIC program lines
RENUMBER	Renumbers your BASIC program, including all GOTOs and GOSUBs
DELETE	Removes groups of BASIC program lines
FIND	Locates and displays the BASIC program lines that contain a specified string
APPEND	Adds a previously SAVEd program to the one currently in your PET
DUMP	Displays the names and values of all the variables used by your program (excluding arrays)
HELP	If your program stops due to an error, HELP displays the offending line and where the PET detected the error.
TRACE	As a program runs, the last six line numbers being executed are shown in the upper right corner of the PET's screen.
STEP	Executes one BASIC line and stops. Pressing SHIFT executes the next line. The line number is displayed in the upper right corner of the screen
OFF	Turns TRACE or STEP off

For the new 16K and 32K PETS, the tool kit consists of a single ROM chip which plugs into the left most empty socket inside the PET. Price £55 plus VAT.
For 8K and other 'old ROM' PETs a small printed circuit board is attached to **the memory expansion and 2nd cassette ports of the PET. Price £75 plus VAT.** Also available for 8K PETS with new ROMS. Please state configuration when ordering.

ACT Petsoft Telephone: 021-455 8585 Telex: 339396
Radclyffe House, 66-68 Hagley Road, Edgbaston, Birmingham B16 8PF

OFFICIALLY APPROVED BY COMMODORE

Duplicating Cassettes for Commodore

One or two of you have expressed concern in the past about the quality of the pre-recorded cassettes that we send out from Commodore. Well, your worries are over. Cassette duplicating is now being done for Commodore by a company called Audiogenic from Berkshire. The quality of the first batch received from them has been superb, and as from January 1st, all tapes from us will have been recorded by Audiogenic.

Their Director, Martin Maynard, takes up the story

Before going into detail, may I introduce myself as Martin Maynard of Audiogenic Limited situated in Reading. My company was called in to duplicate and package the cassette based software produced by Commodore. Although Audiogenic has duplicated music cassettes for the past five years, digital tapes are something new to us, and fortunately, I was able to call upon my past experience in the data-communications industry. The cassette deck containing moving parts, is probably the weakest part of any microcomputer system, and likely to be the first area upon which suspicion will fall when difficulties in loading a programme are experienced. With this in mind, it will be of interest to PET users to know how the signal is recorded on tape and to what lengths Audiogenic goes to, to ensure the entire range of PET pack Softwre will load first time.

The PET cassette deck uses an unequalised recording method to place data on tape, by switching the directin of current through the record head saturating the tape either negatively or positively. The encoding scheme uses three distinct full cycle pulses (see fig. 1), a data zero or one is represented by a pairing of a short and a long pulse. If the shorter pulse is first, the pair is considered a one. The byte market provides reference for byte identification. (See fig. 2). In the playback circuit the recorded signal passes through equalisation and squaring circuits, thus logic level signals are presented to the PET. The PET measures between negative going edges of signals and decodes the data from these measurements.

When we are duplicating tapes, we have to be very conscious of the slew rate of our negative going pulses, as it will be seen that if the slew rate is slow, there is a larger area of indecision presented to the squaring circuits whose threshold is set about the zero crossing point of magnetic flux on tape. After tapes have been duplicated, they are subject to quality control procedure, every one in six tapes is checked using Commodore's "Tape Graph 21S". This programme is loaded in the normal method and when running, examines a signal being received on cassette port one, and measures the timing between negative going pulses and displays the results on the screen as a bar graph (See fig 3), showing up any nasties that may be occuring, with the timing due to uneven tape speed, dirt in the capstan, dirty head and spurious noises, all of which affect the timing on a tape.

Out of each batch of sixty tapes one tape has all its programmes loaded, and the "PRINT PEEK (630) ST" test is used. (See owner's Manual), this must give a 0-0 reply. If all the samples pass this test that batch of tapes is visually inspected and packaged. If any failures are detected, the entire batch is rejected. Listening to a PET tape on your hi-fi set also revealing, lack of high frequency content will indicate dirty or magnetised heads, and speed variations can be detected as overall pitch drifting.

It should be stressed that whilst we take great care in producing "Work first time Software", the system is only as goodfd as its weakest link, which is the cassette deck, and the user should clean and demagnetise both the heads on the deck at least every five hours of operation. Heads are best cleaned with cotton buds and alcolhol, and the capstan pinch roller should be cleaned. Do not rely on TDK cassette demagnetisers as they do not degauss the erase head. When making your own tapes on a PET always use quality audio tapes as budget tapes suffer from drop out and mechanisms fail to run smoothly. Any of the leading makes of low noise tapes are acceptable or of course, official Commodore blank tapes can be purchased from our dealers.

by Martin Maynard
Audiogenic Limited
34/36 Crown Street
Reading
Berks

Figure 1

Data Timing

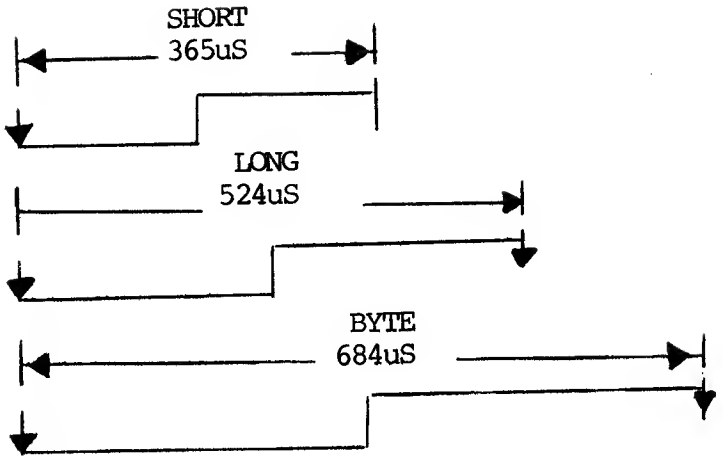


Figure 2

Data In/Out

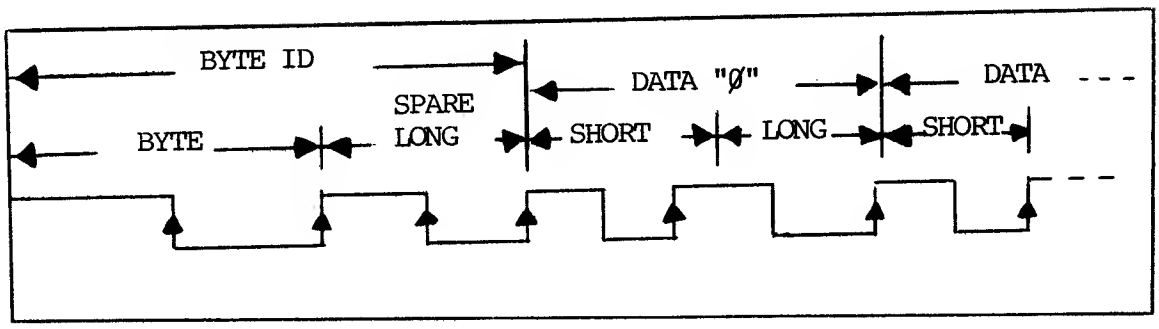
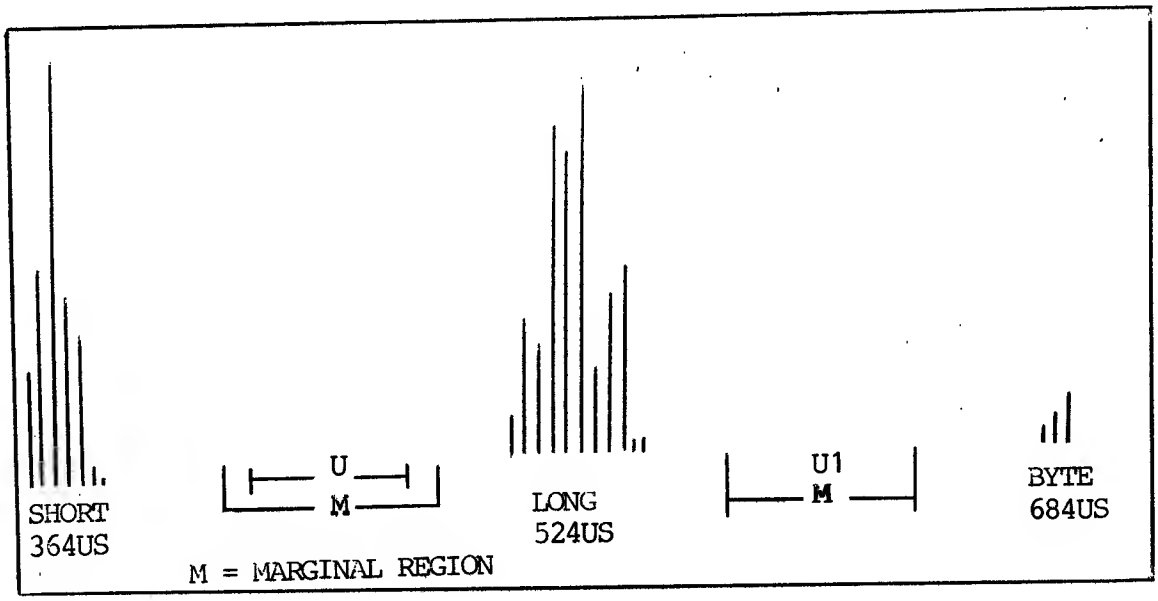


Figure 3



Job Vacancies

A) Course Organiser
($5500 upwards)

B) Administrative Assistant
($4000 upwards)

Commodores' Training Department is expanding. Two extra members of staff are required to join us as soon as possible in the New Year. Brief job descriptions are given below. If you are interested contact Chris Punter on 0753 74111 or write to her at Commodore Systems Division, 818 Leigh Road, Trading Estate, Slough, Berks. She will send you an application form together with a full job specification.

A) Training Course Organiser

To share responsibility for running the BASIC and DISC UTILISATION courses already successfully developed. The job, based at Slough, involves teaching, supervision of students' work and further course development. Part of the job involves keeping up to date with all the latest developments in Micro Computer techniques and time and computing facilities are made available for this purpose. Later, you will be involved in the development and improvement of other Commodore courses and seminars.

B) Training Department Administrative Assistant

To look after the administrative and clerical requirements of the Training Department. The job, based at Slough, includes organising course bookings and venues, acquiring and transport of equipment, maintaining records and producing summaries, liaising with other departments, dealers and customers, normal filing and typing. You will be aided by computer equipment programmed to help rather than deter you in such areas as mailing lists, accounting and word-processing and any necessary training will be given.

The Original Cassette Magazine for the Commodore PET

CURSOR

CURSOR — The cassette program magazine for PET owners. Mailed to you by first class post, each issue contains a dynamic graphic cover, table of contents and at least five new programs. There is a featured game which might cost £8 elsewhere, plus tutorials, programming aids and business routines, and of course CURSOR Notes with news and equipment reviews.

U.K.: £36 for one year subscription (10 issues)
Overseas airmail: £45 for one year.

from...

ACT Petsoft

Radclyffe House, 66-68 Hagley Road, Edgbaston, Birmingham B16 8PF
Telephone: 021-455 8585 Telex: 339396